Directory Server Views

BACKGROUND OF THE INVENTION

TECHNICAL FIELD

The invention relates generally to a directory service. More particularly, the

5 invention relates to a system and method of creating directory server views

and using them to navigate entries in a directory server.

DESCRIPTION OF THE PRIOR ART

A directory service is a structured repository of information about people and

resources within an organization that facilitates management and

10 communication within the organization. On a local area network (LAN) or a

wide area network (WAN), the directory service identifies all aspects of the

network including users, software, hardware, and the various rights and

policies assigned to each. As a result, applications access information without

knowing where a particular resource is physically located and users interact

15 oblivious to the underlying network topology and protocols.

To allow heterogeneous networks to share directory information, the

International Telecommunications Unions (ITU) has proposed a common

structure called "X.500," which is a set of standards covering electronic directory services such as "white pages," "Knowbot," and "whois". However, its complexity and lack of seamless Internet support led to the development of Lightweight Directory Access Protocol (LDAP), which has continued to evolve

5   under the aegis of the Internet Engineering Task Force (IETF).

LDAP has been adopted by many companies, such as America Online Inc. (Netscape Directory Server) and has become a *de facto standard* for directory services. Other LDAP compatible offerings include Novell, Inc.'s Novell Directory Services (NDS) and Microsoft Corporation's Active Directory. The

10   Netscape and Novell products are available for Windows NT and Unix platforms. Novell Directory Services also run on Novell platforms. Microsoft Corporation's Active Directory is an integral part of Microsoft's Windows 2000 and, although it can interface with directory services running on other systems, it is not available for other platforms.

15   Directory service has been widely used in various areas because it provides a standard way to add, update, and retrieve information entries stored in a central or distributed storage place. For example, using LDAP, any client compliant to the LDAP standard can communicate with any directory server compliant to the same LDAP standard.

There is, however, a problem in the deployment of a directory server in that a user is limited to choose either a flat directory information tree or a hierarchical directory information tree. On the one hand, if the user chooses to create a flat directory information tree, it becomes inconvenient to navigate

5   the entries in the directory. On the other hand, if the user chooses to create a hierarchical directory information tree that is useful in navigating the directory, it becomes inconvenient to respond to a need to change the hierarchy.

It would be advantageous to provide a directory server that can create arbitrary hierarchies that are mapped to entries without the need to move the

10   entries in the directory.

It would also be advantageous to keep any changes to the directory server structure to be transparent to any client of the directory server.

It would also be advantageous to facilitate a structured directory information tree in the directory server.

15   **SUMMARY OF THE INVENTION**

The preferred embodiment of the invention provides a technique for hierarchically navigating entries in a way that does not require the physical

3

existence of the entry in any particular place, and which uses information about the entries to place them in the views hierarchy. Given a set of directory views, a user can navigate using one of the directory views that makes the most sense to get to the entries required.

5   In one preferred embodiment of the invention, a method for navigating entries hierarchically in a directory server by creating and using directory views is illustrated. The entries do not need to be physically in any particular place. The directory server may have a flat directory information tree. The existence of the directory views is transparent to a client of the directory server and the

10   client is not required to have special knowledge of the directory views to use them.

Each of the directory views begins with an ordinary entry. The presently preferred method comprises the steps of:

- Creating one or more directory views;

15   - Organizing the directory views into a hierarchy using information about the entries; and

- Using one of the directory views that is best suited for navigating to the entries.

Each of the directory views belongs to a specific object class that contains a filter attribute. This filter attribute contains a filter that describes the views. The filter attribute may be omitted from the views to facilitate a hierarchical directory structure. Each of the directory views may have sub-views which provide a subset of the views. The sub-views may have different subject domains from the directory views.

In another preferred embodiment, a method to search the directory views in the directory server by rewriting filter is provided. The method comprises the steps of:

- Rewriting the given filter to be a sub-tree search of the parent of the top most view in the view hierarchy; and

- Performing the sub-tree search with the rewritten filter.

In this method, the directory entries do not need to be physically in any particular place. The directory server may have a flat directory information tree. The existence of the views is transparent to a client of the directory server and the client requires no special knowledge of the views to use them. Each of the directory views begins with an ordinary entry. Each of the directory views belongs to a specific object class that contains a filter attribute. This filter attribute contains a filter that describes the directory views. The filter

5

attribute may be omitted from the directory views to facilitate a hierarchical directory structure. Each of the directory views may have sub-views which provide a subset of the views. The sub-views may have different subject domains from the directory views.

5   In some embodiments, the rewriting step in the above illustrated method further comprises the steps of:

(a) Collecting filters from the view and all ancestor views of the view to form a first sub-filter;

(b) If the search is not a sub-tree search, collecting all filters from all

10   descendent views to form a second sub-filter;

(c) Adding a third sub-filter to ensure all children of the view are included in the search for one level search or ensure all descendents of the view are included for a sub-tree search; and

(d) Combining the sub-filters from steps (a)-(c) and the given filter to

15   produce the rewritten filter.

Step (a) further comprises the steps of:

(1) Starting from the top most view and working down;

(2) Adding each filter to the first sub-filter in step (a) using the logical AND operator; and

(3) Moving down the hierarchy and going to step (2) until at the view.

Step (b) further comprises the steps of:

(1) Working down the hierarchy until the hierarchy ends;

(2) Adding each filter to the second sub-filter in step (b) using the logical AND operator and the logical NOT operator; and

(3) Repeating step (2) until all sub-views of the view have been accounted for.

Step (c) further comprises the steps of:

(1) For sub-tree searches, using the logical OR operator and a filter which includes the components of the descendent views' distinctive attributes, and which excludes the distinctive attribute of the view;

(2) For one level searches, using the logical OR operator and a filter which includes the components of the children views' distinctive attributes, and which excludes the relative distinctive attribute of all children views of the view using the logical NOT operator; and

7

(3) For base searches, using the filter "objectclass=nsview", wherein "nsview" is the object class of the views.

Step (d) further comprises the steps of:

(1) Combining the third sub-filter from step (c) with the given search filter using the logical AND operator;

(2) Combining the first sub-filter from step (a) and the second sub-filter from step (b) with the given search filter using the logical AND operator;

(3) Combining the resulting filters from steps (1) and (2) using the logical operator OR.

The sub-filters from steps (a), (b) and (c) may be cached so that the filter rewriting only needs to perform step (d), which amounts to simple filter concatenation.

In another preferred embodiment of the invention, a directory server for managing directory information is provided. The directory server comprises:

- A plurality of directory entries; and

- A set of directory views to facilitate hierarchical navigation of the directory entries.

8

The directory entries do not need to be physically in any particular place. The directory server may further comprise a flat directory information tree. It may further comprise a means to search the directory views by rewriting filters. The existence of the directory views is transparent to a client of the directory

5   server and the client is not required to have special knowledge of the directory views to use them. Each of the directory views begins with an ordinary entry. Each of the directory views belongs to a specific object class that contains a filter attribute. The filter attribute containing a filter that describes the directory views. The filter attribute may be omitted from the views to facilitate a

10 hierarchical directory structure. Each of the directory views may have sub-views that provide a subset of the directory views. The sub-views may have different subject domains from the directory views.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating a sample directory view hierarchy;

15 FIG. 2 is a flowchart illustrating the steps to rewrite filter when searching a view;

FIG. 3 is a flowchart illustrating details of step 201 in FIG. 2;

9

FIG. 4 is a flowchart illustrating details of step 202 in FIG. 2;

FIG. 5 is a flowchart illustrating details of step 203 in FIG. 2; and

FIG. 6 is a flowchart illustrating details of step 204 in FIG. 2.

DETAILED DESCRIPTION OF THE INVENTION

5

A directory server view is a technique to navigate entries hierarchically in a manner which does not require the physical existence of the entry in any particular place, and which uses information about the entries to place them in the views hierarchy. When a set of views are realized on a directory, a user

10 can navigate using one of the views which is most appropriate to get to the entries that are required.

For example, if the user's target entries are those which dwell in Mountain View, then the view that is most appropriate is perhaps the one that starts out navigating down using location-based information. If the target entry is an

15 organizational question, however, the organization view may be the best. Both views exist in the directory server at the same time and operate on the same

set of entries. They have different concerns when displaying their version of the directory structure.

Therefore, the directory server views can be regarded as a virtual directory information tree. When a directory server is deployed with the directory views

5   of the invention, any directory structure other than a flat directory information tree is unnecessary.

One way to create a set of hierarchies is to use the innate ability of the directory to use them. In the following preferred embodiment, the features inherent in the directory to realize views, namely its hierarchical directory

10   information tree and its ability to search by filter are used.

Each view begins with an ordinary entry such as an "nsContainer" or "organizationalunit" object. The entry has certain required characteristics including (1) the object class ns-view specifying that the entry is a directory view, and (2) the attribute ns-view-filter.

15   A view may have sub-views that provide a subset of the parent view. There is no restriction on conventional entries being contained in a view. However, one might consider redirecting entries created in a view to a parent of the top most

11

view, perhaps with certain logic to populate the necessary attributes so that it appears in the view it was intended. To facilitate hierarchical organization, a view may omit the `ns-view-filter` attribute, in which case no view processing occurs for one level searches on that view.

5   Referring to FIG. 1, illustrated is an exemplary directory view hierarchy. It comprises a parent view 101, which has an `ns-view-filter` of "(l=CA)". The parent view 101 contains two sub-views: view 102 which has an `ns-view-filter` of "l=Mountain View", and view 103 which has an `ns-view-filter` of "l=San Jose".

10   The attribute `ns-view-filter` is expected to contain an administrator created filter which describes the view. For example, the top of a location base view may have a value of "c=USA", and an organization based one may have a value of " `(&(dc=netscape)(dc=com))` ".

When a search is performed on the view and before processing begins on the

15   search, the search is re-written. Note that the base of the search must refer to a directory view or no view processing is to occur. The search is re-written to be the parent of the top most view in the view hierarchy. The search type is

re-written to be a sub-tree search. The search filter is re-written to perform the correct search to realize the view.

Referring to FIG. 2, the following steps for the re-writing of the filter are illustrated:

5 • Step 201: Collecting filters from the view and all ancestor views of the view to form a sub-filter.

• Step 202: If the search is not a sub-tree search, collecting all filters from all descendent views to form another sub-filter.

• Step 203: Adding a further filter to ensure all children of the view are included in the search for one level search or adding a further filter to ensure all descendents of the view are included for a sub-tree search.

• Step 204: Combining filters from steps 201-203 and the given filter to produce the rewritten filter.

FIG. 3 further illustrates the steps for processing step 201:

15 • Step 301: Starting from the top most view and working down.

• Step 302: Adding each filter to the sub-filter using the logical AND operator.

• Step 303: Moving down the hierarchy and going to step 302 until at the view.

13

FIG. 4 further illustrates the steps for processing step 202:

- Step 401: Working down the hierarchy until the hierarchy ends.

- Step 402: Adding each filter to the sub-filter using the logical AND operator and the logical NOT operator.

5 - Step 403: Repeating step 402 until all sub-views of the view have been accounted for.

FIG. 5 further illustrates the steps for processing step 203:

- Step 501: For sub-tree searches, using the logical OR operator and a filter which includes the components of said descendent views' distinctive
10 attributes (DN), and which excludes the DN of the view.

- Step 502: For one level searches, using the logical OR operator and a filter which includes the components of the children views' DN, and which excludes the RDN of all children view of the view using the logical NOT.

- Step 503: For base searches, using the filter "objectclass=nsview".

15 FIG. 6 further illustrates the steps for processing step 204:

- Step 601: Combining sub-filter from step 203 with the given search filter using the logical AND operator.

- Step 602: Combining sub-filters from steps 201 and 202 with the given search filter using the logical AND operator.

14

- Step 603: Combining the result filters from steps 601 and 602 using the logical operator OR.

It can be seen that using this method, any search can be re-written to conform to a certain view of the directory tree, no matter how that tree is laid out or

5   how complex the view is.

Also note that views are aware of their placement in the directory tree and can be applied to certain branches. For example, if the user is still using a structured directory information tree, a locality based view might be placed under "ou=sales" so that sales people can be found by location.

10   The system design allows for caching of filters which essentially do not change often, if at all, throughout the life of the server invocation. Having cached these filters, there is no need to analyze all descendants of a view to add them to a filter during a search operation. The filters that can be cached include these resulting from steps 201-204. This leaves step 204 as the only

15   work to be done during a search, which amounts to simple filter concatenation, as in known to those skilled in the art.

One distinct feature of the invention is that the directory server views look like any other entry, and their hierarchies look like any other hierarchy. The fact

that a view *is a view* is transparent to the client application and requires no

special knowledge of the views to use them. This attribute means that they

can be put to work in existing installations and can be used by existing custom

clients and third party vendor clients alike, including the command line utilities

5   shipped with the directory server.

Due to the ease of adding new views or re-organizing existing view

hierarchies, a flat directory information tree and a set of views means that

administrators need no longer be concerned by the company reorganizations

and the work they create in the directory.

10   Though all of the examples presented thus far have suggested that sub-views

are similar to a hierarchical drilling down within one subject domain, such as

location, it would be just as feasible to switch the subject of the view. For

instance, a location-based view may at the city level continue on by offering a

choice of views for departments, buildings and maybe even favorite drink.

15   The herein disclosed directory views open up many opportunities to solve

problems for the users and add value to directory server vendors. With the

invention, the user only need a flat directory information tree and a set of

views to facilitate directory navigation, and relieve the problem of users who

have hierarchical directory information trees that must change by adding another view. The invention also offers more than one way to view the directory entries, and thus adds flexibility to working practices when compared to a static model.

5   The invention may be embodied in other automated tools that can be used by anyone desiring to continue interrupted sessions in a stateless network, whether such sessions are business-related, recreational, informational, or otherwise.

Although the invention is described herein with reference to the preferred

10   embodiment, one skilled in the art will readily appreciate that other applications may be substituted for those set forth herein without departing from the spirit and scope of the present invention.

Accordingly, the invention should only be limited by the Claims included below.